APPENDIX A

```
Content-Type: application/octet-stream;
        name="Chart.java"
Content-Transfer-Encoding: quoted-printable
Content-Disposition: attachment;
        filename="Chart.java"
import java.awt.*;
import java.applet.*;
import java.util.Vector;
import java.net.*;
import java.io.*;
import java.util.StringTokenizer;
import java.util.Date;
class Chart extends Panel
{
        public final static int CHART_BAR =3D 1;
        public final static int CHART_LINE =3D 2;
        public final static int CHART_AREA =3D 3;
        public final static int CHART_CANDLE =3D 4;
        public final static int CHART_COMPARE =3D 5;
        public final static int SYMBOL_CHANGED =3D 9001;
        public final static int DETAIL_CHANGED =3D 9002;
        public final static int RIGHTDETAIL_CHANGED =3D 9003;
        public final static int REMOVE_DETAIL =3D 9004;
        public final static int ADD_RECENT =3D 9005;
        private int chartType =3D CHART_BAR;
        private String saveScript =3D "/temp/ImageSave.pl";
=09
        Applet parent;
```

```
private Vector stockHeaders;
int currentSize =3D 0;
int zoom =3D 1;
int chartLeft, chartRight, chartWidth;
int chartTop, chartBottom, chartHeight;
int volumeTop, volumeBottom, volumeHeight;
int scaleMin, scaleMax;
int dateSpan =3D 0;
int minDate, maxDate;
int currentIndex =3D -1;
double valueSpan;
double volumeSpan;
double percentSpan;
double minValue, maxValue;
double minPercent, maxPercent;
boolean usePercent =3D false;
int minVolume, maxVolume;
Image chartImage;
Graphics chartG;
Image offImage;
Graphics offG;
Graphics myG;
Color chartBGColor =3D new Color(204, 204, 153);
Color yAxisColor =3D Color.red;
Color xAxisColor =3D Color.blue;
Color chartBorderColor =3D Color.black;
Color scaleColor =3D Color.lightGray;
Color dragColor =3D Color.blue;
int dateWidth;
```

```
int mouseX =3D -1, mouseY =3D -1;
int dragX =3D -1, dragY =3D -1;
int cursor =3D 1;
boolean mouseInView =3D false;
FontMetrics fm;
int barWidth;
int leftSelectedIndex =3D -1;
int rightSelectedIndex =3D -1;
Font labelFont;
Font titleFont;
boolean dragging =3D false;
Vector drags;
Vector inds;
ADrag ad;
boolean showVolume =3D true;
boolean haveData =3D false;
String holdings =3D null;
String holdingStock =3D null;
private String script =3D "/totalTrader/query.asp?symbol=3D";
private String nameScript =3D "/totalTrader/getName.asp?symbol=3D";
private String symbol =3D "";
private String copyright =3D "Copyright (c) 1998 Prophet Info. = Services, Inc.";
public StockDetail currentDetail =3D null;
public StockDetail currentRightDetail =3D null;
public double currentValue =3D 0;
public String cookieValue =3D null;
public String cookieToGet =3D null;
public String buys =3D null;
public String sells =3D null;
```

```
        private boolean indInVolume =3D false;
        private boolean logChart =3D false;
        boolean forceScale =3D false;
        boolean showSymbol =3D false;
        boolean hideLeft =3D false;
        String legend;
=09
        Chart(Applet sc)
        {
                parent =3D sc;
                stockHeaders =3D new Vector();
                drags =3D new Vector(5, 1);
                inds =3D new Vector(5, 1);
                createImages(800, 600);
                labelFont =3D new Font("Dialog", Font.PLAIN, 10);
                chartG.setFont(labelFont);
                fm =3D chartG.getFontMetrics();
                titleFont =3D new Font("Dialog", Font.BOLD, 16);
                myG =3D getGraphics();
                //resize(parent.size().width, parent.size().height);
        }
        public void createImages(int width, int height)
        {
                if (chartImage !=3D null)
                {
                        chartG.dispose();
                        chartImage =3D null;
                }
                if (offImage !=3D null)
```

```
        {
            offG.dispose();
            offImage =3D null;
        }
        chartImage =3D parent.createImage(width, height);
        chartG =3D chartImage.getGraphics();
=09
        offImage =3D parent.createImage(width, height);
        offG =3D offImage.getGraphics();
    }
    public void destroy()=09
    {=09
        System.gc();
        offG.dispose();
        chartG.dispose();
        offImage =3D null;
        chartImage =3D null;
        System.gc();
    }
    public void setType(int t)
    {
        chartType =3D t;
        updateChart();
        //txtSymbol.requestFocus();
    }
    public void deleteIndicator(int i)
    {
        inds.removeElementAt(i);
        checkShowVolume();
```

```
        updateChart();
    }
    public void checkShowVolume()
    {
        indInVolume =3D false;
        for (int i =3D 0; i < inds.size(); i++)
        {
            Indicator ind =3D (Indicator)inds.elementAt(i);
            if (ind.type =3D=3D Indicator.MACD)
                indInVolume =3D true;
            else if (ind.type =3D=3D Indicator.FAST_STOCHASTIC)
                indInVolume =3D true;
            else if (ind.type =3D=3D Indicator.SLOW_STOCHASTIC)
                indInVolume =3D true;
            else if (ind.type =3D=3D Indicator.RSI)
                indInVolume =3D true;
        }
    }
    public void deleteIndicators()
    {
        inds.removeAllElements();
        indInVolume =3D false;
        updateChart();
    }
    public void addIndicator(Indicator ind)
    {
        inds.addElement(ind);
        checkShowVolume();
```

```
            if (ind.type =3D=3D Indicator.COMPARE)
                loadStock(ind.compare, null, false, currentHeader().duration);
            updateChart();
        }
        public void unZoom()
        {
            try
            {
                setDates(StockDetailAt(0).getDate(), =
StockDetailAt(currentHeader().count() - 1).getDate());
            }
            catch (Exception e) {}
        }
        public void refresh()
        {
            updateChart();
            repaint();
        }
        public void reshape(int x, int y, int  width, int  height)
        {
            super.reshape(x, y, width, height);
            resetDimensions();
        }
        public void setDates(Date left, Date right)
        {
            minDate =3D 0;
            while (StockDetailAt(minDate).getDate().before(left)) minDate++;
            maxDate =3D minDate;
            while (StockDetailAt(maxDate).getDate().before(right) && (maxDate < =
```

```
currentHeader().count())) maxDate++;
                dateSpan =3D Math.max(maxDate - minDate + 1, 1);
                resetDimensions();
                resetScale();
                updateChart();
        }
        public void hideLeft()
        {
                hideLeft =3D true;
                updateChart();
        }
        public void showLeft()
        {
                hideLeft =3D false;
                updateChart();
        }
        public void resetDimensions()
        {
                try
                {
                        int availableHeight =3D size().height;
                        int labelHeight =3D fm.getHeight() * 2;
                        dateWidth =3D fm.stringWidth("99/99/99 ");
                        chartLeft =3D fm.stringWidth("999.999");
                        chartRight =3D Math.min(size().width, 1200) - chartLeft;
                        if (hideLeft)
                                chartLeft =3D 5;
                        chartWidth =3D chartRight - chartLeft;
                        chartTop =3D 0;
```

```
                volumeHeight =3D 0;
                if (showVolume)
                        volumeHeight =3D (availableHeight >> 2);
            chartBottom =3D Math.min(availableHeight, 800) - volumeHeight - =
(labelHeight >> 1);
                volumeTop =3D chartBottom + 14;
            volumeBottom =3D Math.min(availableHeight, 800) - (labelHeight >> 1)
=
- 3;
                volumeHeight =3D volumeBottom - volumeTop;
                chartHeight =3D chartBottom - chartTop;
                minValue =3D 1;
                maxValue =3D 100;
                minVolume =3D 1;
                maxVolume =3D 100;
                valueSpan =3D maxValue - minValue;
                volumeSpan =3D maxVolume - minVolume;
            }
            catch (Exception e) { } //may break if there is no data
        }
        public void resetScale()
        {
            minValue =3D Integer.MAX_VALUE;
            maxValue =3D Integer.MIN_VALUE;
            minPercent =3D Integer.MAX_VALUE;
            maxPercent =3D Integer.MIN_VALUE;
            minVolume =3D 0;
            maxVolume =3D Integer.MIN_VALUE;
            findExtremes(null);
```

```
usePercent =3D false;
for (int count =3D 0; count < inds.size(); count++)
{
        Indicator ind =3D (Indicator)inds.elementAt(count);
        if (ind.type =3D=3D Indicator.COMPARE)
        {
                usePercent =3D true;
                findExtremes(ind.compare);
        }
}
=09
if (chartType =3D=3D Chart.CHART_COMPARE)
        usePercent =3D true;
/*maxValue =3D (int)(maxValue + 0.9999);
minValue =3D (int)(minValue - 0.9999);
*/
if ((minValue <=3D 2) && (haveData) && (!forceScale))
{
        logChart =3D false;
        postEvent(new Event(this, 4010, null));
}
double oldMin =3D minValue;
if ((minValue < 0.5) && (!logChart))
        minValue =3D 0;
if (logChart && (minValue =3D=3D 0))
        minValue =3D Math.max(0.001, oldMin);
if (maxVolume > 0)
{
        int numbers =3D ("" + maxVolume).length();
```

```
                int firstDigit =3D Integer.parseInt(""+maxVolume).substring(0,
1));
                maxVolume =3D (int)Math.pow(10, numbers - 1) * (firstDigit +
1);
        }

        /*minPercent =3D (minValue - zeroPercent()) / zeroPercent() - 0.1;
        maxPercent =3D (maxValue - zeroPercent()) / zeroPercent() + 0.1;*/
        maxPercent +=3D 1;
        minPercent +=3D 1;
        percentSpan =3D maxPercent - minPercent;
        valueSpan =3D maxValue - minValue;
        volumeSpan =3D maxVolume - minVolume;
    }
    public void findExtremes(String s)
    {
        double zero =3D -1;
        {
            for (int i =3D minDate; i < maxDate+1; i++)
            {
                StockDetail sd;
                if (s =3D=3D null)
                    sd =3D StockDetailAt(i);
                else
                    sd =3D StockDetailAt(s, i);
                if (sd !=3D null)
                {
                    if (zero =3D=3D -1)
                        zero =3D sd.getClose();
```

```
                        if (sd.getHigh() > maxValue)
                                maxValue =3D sd.getHigh() + 0.1;
                        if (sd.getLow() < minValue)
                                minValue =3D sd.getLow();
                        if (sd.getVolume() > maxVolume)
                                maxVolume =3D sd.getVolume();

                        double highPercent =3D (sd.getHigh() - zero) / zero;
                        double lowPercent =3D (sd.getLow() - zero) / zero;
                        if (highPercent > maxPercent)
                                maxPercent =3D highPercent;
                        if (lowPercent < minPercent)
                                minPercent =3D lowPercent;
                    }=20
                    //else
                    //      System.out.println("out of date: " + s + " " + i);
                }
=09
        }
}
public double zeroPercent(String sym)
{
        try
        {
                int offset =3D 0;
                Date d =3D StockDetailAt(minDate).getDate();
                while (StockDetailAt(sym, offset).getDate().before(d))
                        offset++;
                return StockDetailAt(sym, offset).getClose();
```

```
            }
            catch (Exception e) { }
            return 1;
        }
        public double zeroPercent()
        {
            try
            {
                return StockDetailAt(minDate).getClose();
            }
            catch (Exception e) { }
            return 0;
        }
        public void drawTransactions(String s, int t)
        {
            if (s =3D=3D null)
                return;
            chartG.setColor(Color.red.darker());
            StringTokenizer st =3D new StringTokenizer(s, ";");
            while (st.hasMoreTokens())
            {
                StringTokenizer st2 =3D new StringTokenizer(st.nextToken(), ",");
                double value =3D new Double(st2.nextToken()).doubleValue();
                Date date;
                date =3D new Date(st2.nextToken());
                int x =3D dateToX(date) + (barWidth >> 1);
                int y =3D valueToY(value);
                if ((x >=3D chartLeft) && (x <=3D chartRight) &&
                    (y >=3D chartTop) && (y <=3D chartBottom))
```

```
                {
                    if (t =3D=3D 0)
                        chartG.drawOval(x-3, y-3, 6, 6);
                    else
                        chartG.fillOval(x-3, y-3, 6, 6);
                }
            }
        }
        public void updateChart()
        {
            if ((chartImage =3D=3D null) || (size().width > = offImage.getWidth(this))
                ||
                    (size().height > offImage.getHeight(this)))
                    createImages(size().width, size().height);
            resetDimensions();
            resetScale();
            chartG.setFont(labelFont);
            chartG.setColor(chartBGColor);
            chartG.fillRect(0, 0, size().width, size().height);
            chartG.setColor(Color.white);
            chartG.fillRect(chartLeft, chartTop, chartWidth, chartHeight);
            chartG.fillRect(chartLeft, volumeTop, chartWidth, volumeHeight);
            chartG.setColor(chartBorderColor);
            chartG.drawRect(chartLeft, chartTop, chartWidth, chartHeight);
            chartG.drawRect(chartLeft, volumeTop, chartWidth, volumeHeight);
            if (!haveData)
            {
                repaint();
                return;
```

```
                }
        =09
                barWidth =3D Math.max(1, (int)(chartWidth / Math.max(dateSpan, 1)));
                if ((holdings !=3D null) && =
5       (currentHeader().symbol.equals(holdingStock)))
                {
                        try
                        {
                                chartG.setColor(new Color(204, 204, 255));
10                      StringTokenizer st =3D new StringTokenizer(holdings, ",");
                                while (st.hasMoreTokens())
                                {
                                        Date d1, d2;
                                        d1 =3D new Date(st.nextToken());
15                                      if (st.hasMoreTokens())
                                                d2 =3D new Date(st.nextToken());
                                        else
                                                d2 =3D new Date("12/12/2100");
                                        int x1 =3D dateToX(d1);
20                                      int x2 =3D dateToX(d2) + barWidth;
                                        if (x1 < chartLeft)
                                                x1 =3D chartLeft;
                                        if (x2 >=3D chartRight)
                                                x2 =3D chartRight - 1;
25                                      chartG.fillRect(x1, chartTop+1, x2 - x1,
        chartHeight-1);
                                }
                        }
                        catch (Exception e)
```

```
            {
                System.out.println(""+e);
            }
            drawAxis(true);
            drawTransactions(buys, 0);
            drawTransactions(sells, 1);
        } else
            drawAxis(true);
        int hb =3D barWidth >> 1;
        int qb =3D barWidth >> 2;
        int eb =3D barWidth >> 3;
        int zero =3D (int)valueToY(minValue);
        int lastX =3D dateToX(minDate);
        int lastY =3D valueToY(StockDetailAt(minDate).getClose());
        int lastLeft =3D dateToX(minDate);
        chartG.setColor(Color.black);
        int theMaxDate =3D maxDate;
        int tempChart =3D chartType;
        if (usePercent)
            tempChart =3D CHART_COMPARE;
        if ((symbol.length() =3D=3D 5) && (symbol.endsWith("X")))
            tempChart =3D CHART_LINE;
        switch (tempChart) {
        case CHART_COMPARE:
            try
            {
                lastY =3D percentToY(1);
                for (int i =3D minDate; i <=3D theMaxDate; i++)
                {
```

```
                        StockDetail sd =3D StockDetailAt(i);
                        sd.x1 =3D dateToX(i);
                        sd.x2 =3D sd.x1 + barWidth;
                        int y;
                        y =3D percentToY(sd.getClose() / zeroPercent());
                        if (sd.x2 >=3D chartRight)
                                sd.x2 =3D chartRight;
                        if (y >=3D chartBottom)
                                y =3D chartBottom;
                        chartG.drawLine(lastX, lastY, sd.x2, y);
                        lastX =3D sd.x2;
                        lastY =3D y;
                }       =09
        } catch (Exception e) {}
        break;
case CHART_LINE:
        try
        {
                for (int i =3D minDate; i <=3D theMaxDate; i++)
                {
                        StockDetail sd =3D StockDetailAt(i);
                        sd.x1 =3D dateToX(i);
                        sd.x2 =3D sd.x1 + barWidth;
                        int y;
                        y =3D valueToY(sd.getClose());
                        if (sd.x2 >=3D chartRight)
                                sd.x2 =3D chartRight;
                        if (y >=3D chartBottom)
                                y =3D chartBottom;
```

```
                    chartG.drawLine(lastX, lastY, sd.x2, y);
                    lastX =3D sd.x2;
                    lastY =3D y;
                }   =09
            } catch (Exception e) {}
            break;
        case CHART_AREA:
        =09
            try
            {
                for (int i =3D minDate; i <=3D theMaxDate; i++)
                {
                    StockDetail sd =3D StockDetailAt(i);
                    sd.x1 =3D dateToX(i);
                    sd.x2 =3D sd.x1 + barWidth;
                    int y =3D valueToY(sd.getClose());
                    if (sd.x2 >=3D chartRight)
                        sd.x2 =3D chartRight;
                    if (y >=3D chartBottom)
                        y =3D chartBottom;
                    Polygon pg =3D new Polygon();
                    pg.addPoint(lastX, lastY);
                    pg.addPoint(sd.x2, y);
                    pg.addPoint(sd.x2, zero);
                    pg.addPoint(lastX, zero);
                    chartG.fillPolygon(pg);
                    lastX =3D sd.x2;
                    lastY =3D y;
                }   =09
```

```
        } catch (Exception e) { }
        break;
case CHART_BAR:
        try
        {
                for (int i =3D minDate; i <=3D theMaxDate; i++)
                {
                        StockDetail sd =3D StockDetailAt(i);
                        sd.x1 =3D dateToX(i);
                        sd.x2 =3D sd.x1 + barWidth;
                        int y1 =3D valueToY(sd.getHigh());
                        int y2 =3D valueToY(sd.getLow());
                        if (sd.x2 >=3D chartRight)
                                sd.x2 =3D chartRight;
                        chartG.drawLine(sd.x1 + hb, y1, sd.x1 + hb, y2);
                        int oy =3D valueToY(sd.getOpen());
                        int cy =3D valueToY(sd.getClose());
                        chartG.drawLine(sd.x1, oy, sd.x1 + hb, oy);
                        chartG.drawLine(sd.x2 - hb, cy, sd.x2, cy);
                        lastLeft =3D sd.x2;
                }       =09
        } catch (Exception e) { }
        break;
case CHART_CANDLE:
        try
        {
                for (int i =3D minDate; i <=3D theMaxDate; i++)
                {
                        StockDetail sd =3D StockDetailAt(i);
```

```
                                    sd.x1 =3D dateToX(i);
                                    sd.x2 =3D sd.x1 + barWidth;
                                    int oy =3D valueToY(sd.getOpen());
                                    int cy =3D valueToY(sd.getClose());
                                    int hy =3D valueToY(sd.getHigh());
                                    int ly =3D valueToY(sd.getLow());
                                    if (sd.x2 >=3D chartRight)
                                            sd.x2 =3D chartRight;
                                    int topBar =3D oy;
                                    int bottomBar =3D cy;
                                    if (cy < oy)
                                    {
                                            topBar =3D cy;
                                            bottomBar =3D oy;
                                    }
                                    if (sd.getClose() < sd.getOpen())
            chartG.fillRect(sd.x1 + eb, topBar, sd.x2 - sd.x1 - (2*eb), = bottomBar - topBar);
                                    else
        chartG.drawRect(sd.x1 + eb, topBar, sd.x2 - sd.x1 - (2*eb), = bottomBar - topBar);
                            chartG.drawLine(sd.x1 + hb, hy, sd.x1 + hb, topBar);
                            chartG.drawLine(sd.x1 + hb, bottomBar, sd.x1 + hb, ly);
                                    lastLeft =3D sd.x2;
                                }       =09
                            } catch (Exception e) {}
                            break;
                        }
                =09
                        chartG.setFont(labelFont);
                        chartG.setColor(Color.blue);
```

```
chartG.clipRect(chartLeft+1, chartTop+1, chartWidth-2, chartHeight-1);
int ly =3D chartTop + 2 * chartG.getFontMetrics().getHeight() + 10;
theMaxDate++;
for (int count =3D 0; count < inds.size(); count++)
{
        chartG.dispose();
        chartG =3D chartImage.getGraphics();
        Indicator ind =3D (Indicator)inds.elementAt(count);
        chartG.setColor(Color.black);
        chartG.drawString(ind.toString(), chartLeft + 35, ly);
        chartG.setColor(ind.color);
chartG.fillRect(chartLeft + 10, ly - = (chartG.getFontMetrics().getHeight() >> 1), 20, 5);
        lastX =3D 0;
        lastY =3D 0;
        ly +=3D chartG.getFontMetrics().getHeight();
        if (ind.type =3D=3D Indicator.MOVING_AVERAGE)
        {
                int points =3D (int)ind.value1;
                try
                {
                        for (int i =3D Math.max(points - 1, minDate); i
<=3D theMaxDate; =i++)
                        {
                                int ma =3D valueToY(MA(points, i));
                                if (lastY !=3D 0)
                        chartG.drawLine(lastX, lastY, dateToX(i), ma);
                                lastX =3D dateToX(i);
                                lastY =3D ma;
                        }
```

```
(double)(StockDetailAt(i).getClose() - lowestLow) / =
(double)(highestHigh - lowestLow);
                    int y =3D volumeTop + (int)((volumeHeight-2) * (1-per)) + 1;
                                ys[i] =3D y;
                                if ((lastY >=3D 0) && (ind.type =3D=3D
Indicator.FAST_STOCHASTIC))
                            chartG.drawLine(lastX, lastY, dateToX(i), y);
                                lastX =3D dateToX(i);
                                lastY =3D y;
                            }
                        }
                        catch (Exception e) { }
                        if (ind.type =3D=3D Indicator.FAST_STOCHASTIC)
                chartG.setColor(new Color(255 - ind.color.getRed(), 255 - =
ind.color.getGreen(), 255 - ind.color.getBlue()).darker());
                        lastY =3D -1;
                        lastX =3D 0;
                        double ma_sum =3D 0;
                        int ma_count =3D 0;
                        int newys[] =3D new int[theMaxDate + 1];
                        try
                        {
                            for (int i =3D 0; i < ma_period; i++)
                                ma_sum +=3D ys[i];
                            for (int i =3D ma_period; i <=3D theMaxDate; i++)
                            {
                                int y =3D (int)(ma_sum / ma_period);
                                newys[i] =3D y;
                                if (i > ma_period)
```

```
                    chartG.drawLine(lastX, lastY, dateToX(i), y);
                        ma_sum -=3D ys[i-ma_period];
                        ma_sum +=3D ys[i];
                        lastY =3D y;
                        lastX =3D dateToX(i);
                    }
                }=20
                catch (Exception e) { }
                if (ind.type =3D=3D Indicator.SLOW_STOCHASTIC)
                {
            chartG.setColor(new Color(255 - ind.color.getRed(), 255 - =
ind.color.getGreen(), 255 - ind.color.getBlue()).darker());
                    lastY =3D -1;
                    lastX =3D 0;
                    ma_sum =3D 0;
                    ma_count =3D 0;
                    try
                    {
                        for (int i =3D 0; i < ma_period; i++)
                            ma_sum +=3D newys[ma_period];
                    for (int i =3D ma_period; i <=3D theMaxDate; i++)
                        {
                        int y =3D (int)(ma_sum / ma_period);
                            if (i > ma_period)
                    chartG.drawLine(lastX, lastY, dateToX(i), y);
                            ma_sum -=3D newys[i-ma_period];
                            ma_sum +=3D newys[i];
                            lastY =3D y;
                            lastX =3D dateToX(i);
```

```
                                        }
                                    }=20
                                    catch (Exception e) { }
                                }
                                chartG.setColor(ind.color);
            //chartG.drawLine(chartLeft, volumeTop + (int)(volumeHeight * 0.2), =
chartRight, volumeTop + (int)(volumeHeight * 0.2));
            //chartG.drawLine(chartLeft, volumeTop + (int)(volumeHeight * 0.8), =
chartRight, volumeTop + (int)(volumeHeight * 0.8));
                } else if (ind.type =3D=3D Indicator.RSI)
                {
                    chartG.setColor(ind.color);
                    chartG.clipRect(chartLeft+1, volumeTop+1, chartWidth-2,
=
volumeHeight-1);
                    int points =3D (int)ind.value1;
                    int y =3D 0;
                    try
                    {
        for (int i =3D Math.max(points - 1, minDate); i <=3D theMaxDate; = i++)
                        {
                            double upDays =3D countUp(points, i);
                        double downDays =3D countDown(points, i);
                            double RSI =3D 0;
                            try
                            {
                        RSI =3D 100 - (100.0 / (1 + (upDays / downDays)));
                    y =3D volumeTop + (int) (volumeHeight * (1 - (RSI/100)));
                            }
```

```
                    catch (Exception e) { } //overflow error
                    if (lastY !=3D 0)
                chartG.drawLine(lastX, lastY, dateToX(i), y);
                    lastX =3D dateToX(i);
                    lastY =3D y;
                }
            }
            catch (Exception e) { }
            chartG.dispose();
            chartG =3D chartImage.getGraphics();
        } else if (showVolume && (ind.type =3D=3D Indicator.MACD))
        {
            int period1 =3D (int)ind.value1;
            int period2 =3D (int)ind.value2;
            double lastEMA1 =3D StockDetailAt(0).getClose();
            double lastEMA2 =3D StockDetailAt(0).getClose();
            chartG.setColor(ind.color);
            double emas[] =3D new double[theMaxDate+1];
            double maxDiff =3D -99999;
            double minDiff =3D +99999;
            try
            {
                //FIND MAX/MIN AND ALL THE DIFFS
                for (int i =3D 0; i <=3D theMaxDate; i++)
                {
            double newEMA1 =3D EMA(period1, i, lastEMA1);
            double newEMA2 =3D EMA(period2, i, lastEMA2);
                    emas[i] =3D newEMA1 - newEMA2;
                    maxDiff =3D Math.max(emas[i], maxDiff);
```

```
                    minDiff =3D Math.min(emas[i], minDiff);
                    lastEMA1 =3D newEMA1;
                    lastEMA2 =3D newEMA2;
                }
            }
            catch (Exception e) { }
            double diffSpan =3D maxDiff - minDiff;
chartG.clipRect(chartLeft+1, volumeTop+1, chartWidth-2, = volumeHeight-1);
            try
            {
                //NOW, PLOT THE DIFFS
                for (int i =3D 0; i <=3D theMaxDate; i++)
                {
int y =3D volumeTop + (int)(volumeHeight * (1 - ((emas[i] + = Math.abs(minDiff)) /
diffSpan)));
                    emas[i] =3D y;
                if ((i > 0) && (i >=3D Math.max(period1, period2) - 1))
                chartG.drawLine(lastX, lastY, dateToX(i), y);
                    lastX =3D dateToX(i);
                    lastY =3D y;
                }
            }
            catch (Exception e) { }
        =09
            //plot the signal line
            int period =3D 9;
            lastY =3D -1;
            lastX =3D 0;
            double ma_sum =3D 0;
```

```
int ma_count =3D 0;
chartG.setColor(chartG.getColor().brighter());
try
{
    for (int i =3D 1; i <=3D theMaxDate; i++)
    {
        ma_sum +=3D emas[i];
        ma_count++;
        if (ma_count >=3D period)
        {
    int y =3D (int)((double)(ma_sum / period));
    if ((lastY >=3D 0) && (i % 2 =3D=3D 0))
    chartG.drawLine(lastX, lastY, dateToX(i), y);
            ma_sum -=3D emas[i-period];
            lastY =3D y;
            lastX =3D dateToX(i);
        }
    }
}=20
    catch (Exception e) { }
} else if (showVolume && (ind.type =3D=3D Indicator.EMA))
{
    int period =3D (int)ind.value1;
    double lastEMA =3D StockDetailAt(0).getClose();
    chartG.setColor(ind.color);
    try
    {
for (int i =3D Math.max(0, minDate - period); i <=3D theMaxDate; = i++)
    {
```

```
                    double newEMA =3D EMA(period, i, lastEMA);
                        int ema =3D valueToY(newEMA);
                        lastEMA =3D newEMA;
                        if ((lastY !=3D 0) && (i >=3D period - 1))
                    chartG.drawLine(lastX, lastY, dateToX(i), ema);
                        lastX =3D dateToX(i);
                        lastY =3D ema;
                    }
                }
                catch (Exception e) { }
            } else if (ind.type =3D=3D Indicator.COMPARE)
            {
                chartG.setColor(ind.color);
                try
                {
                    lastX =3D dateToX(0);
                    lastY =3D percentToY(1);
                    double zeroPercent =3D zeroPercent(ind.compare);
                    for (int i =3D minDate; i <=3D theMaxDate; i++)
                    {
                StockDetail sd =3D StockDetailAt(ind.compare, i);
                        sd.x1 =3D dateToX(i);
                        sd.x2 =3D sd.x1 + barWidth;
                        int y;
                    y =3D percentToY(sd.getClose() / zeroPercent);
                        if (sd.x2 >=3D chartRight)
                            sd.x2 =3D chartRight;
                        if (y >=3D chartBottom)
                            y =3D chartBottom;
```

```
                                        if (lastX >=3D chartLeft)
                                        chartG.drawLine(lastX, lastY, sd.x2, y);
                                        lastX =3D sd.x2;
                                        lastY =3D y;
                                }       =09
                        } catch (Exception e)=20
                        {
                                System.out.println(e);
                        }
                }
                chartG.dispose();
                chartG =3D chartImage.getGraphics();
                if ((currentHeader().splits !=3D null) && !usePercent)
                {
                        StringTokenizer st =3D new
StringTokenizer(currentHeader().splits, = ",");
                        while (st.hasMoreTokens())
                        {
                                Date d =3D new Date(st.nextToken());
                                String s =3D st.nextToken();
                                int x =3D dateToX(d) + hb;
                                int w =3D 5;
                                if (x >=3D chartLeft && x <=3D chartRight)
                                {
                                        StockDetail sd =3D StockDetailAt(xToIndex(x));
                                        int y =3D valueToY(sd.getClose());
                                        Polygon pg =3D new Polygon();
                                        pg.addPoint(x, y);
```

```
                                pg.addPoint(x-w, y-2*w);
                                pg.addPoint(x+w, y-2*w);
                                chartG.setColor(Color.yellow);
                                chartG.fillPolygon(pg);
                                chartG.setColor(Color.black);
                                chartG.drawPolygon(pg);
                        }
                }
        }
        if ((showVolume) && (!indInVolume))
        {
chartG.clipRect(chartLeft+1, volumeTop+1, chartWidth-2, = volumeHeight-1);
                int vb =3D volumeToY(0);
                chartG.setColor(Color.blue);
                try
                {
                        for (int i =3D minDate; i <=3D theMaxDate; i++)
                        {
                                StockDetail sd =3D StockDetailAt(i);
                                int y =3D volumeToY(sd.getVolume());
                                int x =3D dateToX(i);
                                chartG.fillRect(x + qb, y, Math.max(hb, 1), vb - y);
                        }       =09
                } catch (Exception e) {}
                if (currentHeader().isFuture())
                {
System.out.println("drawing future openint" + = currentHeader().maximumOpenint);
                        chartG.setColor(Color.red.darker());
                        int oldX =3D -1;
```

```
                    int oldY =3D -1;
        int intSpan =3D currentHeader().maximumOpenint - =
currentHeader().minimumOpenint;
                    for (int i =3D minDate; i <=3D theMaxDate; i++)
                    {
                        StockDetail sd =3D StockDetailAt(i);
                        if (sd !=3D null)
                        {
                            int y =3D openIntToY(sd.getOpenint());
                            int x =3D dateToX(i);
                        if ((oldX >=3D 0) && (sd.getOpenint() !=3D 0))
                                chartG.drawLine(oldX, oldY, x, y);
                            oldX =3D x;
                            oldY =3D y;
                        }
                    }       =09
                }
                chartG.dispose();
                chartG =3D chartImage.getGraphics();
            }=09
            //3d borders
            chartG.setColor(chartBorderColor);
            chartG.drawRect(chartLeft, chartTop, chartWidth, chartHeight);
            chartG.drawRect(chartLeft, volumeTop, chartWidth, volumeHeight);
            chartG.setColor(chartBorderColor.brighter());
            chartG.fillRect(chartLeft+4, chartBottom+1, chartWidth, 3);
            chartG.fillRect(chartRight+1, chartTop+3, 3, chartHeight);
            chartG.fillRect(chartLeft+4, volumeBottom+1, chartWidth, 2);
            chartG.fillRect(chartRight+1, volumeTop+2, 3, volumeHeight);
```

```
                chartG.clipRect(chartLeft+1, chartTop+1, chartWidth-2, chartHeight-2);
                for (int i =3D 0; i < drags.size(); i++)
                {
                        ADrag ad =3D (ADrag)drags.elementAt(i);
                        chartG.setColor(dragColor);
                        int x1 =3D dateToX(ad.getDate1()) + hb;
                        int x2 =3D dateToX(ad.getDate2()) + hb;
                        int y1 =3D valueToY(ad.getValue1());
                        int y2 =3D valueToY(ad.getValue2());
                        chartG.drawLine(x1, y1, x2, y2);
                }
                chartG.dispose();
                chartG =3D chartImage.getGraphics();
                chartG.setColor(Color.black);
                chartG.setFont(new Font("Dialog", Font.PLAIN, 8));
                chartG.drawString(copyright, chartLeft + 5, chartBottom - 6);
                repaint();
        }
        public double EMA(int period, int x, double lastEMA)
        {
                double percentage =3D 2.0 / (period + 1.0);
                double currentClose =3D StockDetailAt(x).getClose();
                //System.out.println(""+currentClose + " " + percentage);
                return (double)(percentage*currentClose) + (double)((1-percentage) * =
lastEMA);
        }
        public double countUp(int period, int start)
        {
                double sum =3D 0;
```

```
        for (int i =3D start - period + 1; i < start + 1; i++)
                if (StockDetailAt(i).getClose() >=3D StockDetailAt(i).getOpen())
                sum +=3D StockDetailAt(i).getClose();
        return sum;
}
public double countDown(int period, int start)
{
        double sum =3D 0;
        for (int i =3D start - period + 1; i < start + 1; i++)
                if (StockDetailAt(i).getClose() < StockDetailAt(i).getOpen())
                sum +=3D StockDetailAt(i).getClose();
        return sum;
}
public double MA(int period, int start)
{
        double sum =3D 0;
        for (int i =3D start - period + 1; i < start + 1; i++)
                sum +=3D StockDetailAt(i).getClose();
        return sum / period;
}
public double EMA(double SF, double lastEMA, int start)
{
        double Xt =3D StockDetailAt(start).getClose();
        return lastEMA + SF * (Xt - lastEMA);
}
public void drawAxis(boolean lines)
{
        drawYAxis(lines);
        drawXAxis(lines);
```

```
}
public void drawYAxis(boolean lines)
{
        //DRAW Y AXIS LABELS
        int steps;
        double yStep, yValue;
        steps =3D (int)((chartHeight - 30) / fm.getHeight());
        if (usePercent)
        {
                yStep =3D 1.1 * percentSpan / steps;
                yValue =3D minPercent;
        } else  {
                yStep =3D (maxValue - minValue) / steps;
                yValue =3D minValue;
                if (yStep < 0.1)
                        yStep =3D (int)((yStep + 0.01) * 100) / 100.0;
                else if (yStep < 1)
                        yStep =3D (int)((yStep + 0.1) * 10) / 10.0;
                else if (yStep < 10)
                        yStep =3D (int)((yStep + 1) * 1) / 1.0;
        =09
                if ((yStep < 1) && (yStep > 0.5))
                        yStep =3D 1;
                if ((yStep < 0.5) && (yStep > 0.1))
                        yStep =3D 0.5;
                yStep =3D Math.max(yStep, 0.01);
                if (minValue > 10)
                {
                        yValue =3D Math.max((int)yValue, (int)(yValue+0.9999));
```

```
                        if (yValue - 0.5 >=3D minValue)
                                yValue =3D yValue - 0.5;
                        //yStep =3D (int)(yStep + 0.99999);
                }
        }
        int y;
        if (usePercent)
        {
                y =3D percentToY(yValue);
                chartG.setColor(Color.black);
                int zeroLevel =3D percentToY(1);
                chartG.drawLine(chartLeft, zeroLevel, chartRight, zeroLevel);
        }
        else
                y =3D valueToY(yValue);
        while (y > chartTop + (fm.getHeight() >> 1))
        {
                chartG.setColor(yAxisColor);
                double theValue =3D yValue;
                String label;
                if (usePercent)
                {
                        theValue =3D 100 * (yValue - 1);
                        label =3D
("+(Math.abs(theValue)+0.0001)+"00").substring(0, 4);
                        if (theValue < 0)
                                label =3D "-" + label;
                        else if (theValue > 0)
                                label =3D "+"+ label;
```

```
                label =3D label + "%";
            }
            else
                label =3D (""+(theValue+0.00001)+"000").substring(0, 5);
            if (label.indexOf("E") > 0)
                label =3D "0";
            if (!hideLeft)
chartG.drawString(label, chartLeft - fm.stringWidth(label) - 2, y + = fm.getDescent());
chartG.drawString(label, chartRight + fm.stringWidth("9"), y + = fm.getDescent());
            if (lines)
            {
                chartG.setColor(scaleColor);
                chartG.drawLine(chartLeft + 1, y, chartRight - 1, y);
            }
            yValue +=3D yStep;
            if (usePercent)
                y =3D percentToY(yValue);
            else
                y =3D valueToY(yValue);
        }
        int[] volumes =3D new int[5];
        volumes[0] =3D 0;
        volumes[1] =3D maxVolume / 2;
        volumes[2] =3D maxVolume / 4 ;
        volumes[3] =3D 3 * maxVolume / 4;
        volumes[4] =3D maxVolume;
        if (indInVolume)
        {
            volumes[0] =3D 0;
```

```
            volumes[1] =3D 25;
            volumes[2] =3D 50;
            volumes[3] =3D 75;
            volumes[4] =3D 100;
        }
        for (int i =3D 0; i < volumes.length; i++)
        {
            int z;
            String label;
            if (indInVolume)
            {
        z =3D volumeTop + (int)(volumeHeight * (1 - (volumes[i] / 100.0)));
                label =3D "" + volumes[i];
            }
            else
            {
                z =3D volumeToY(volumes[i]);
                int num =3D (int)(volumes[i] / 1000);
                if (num >=3D 1000)
                {
                    int pos =3D (""+num).length() - 3;
        label =3D (""+num).substring(0, pos) + "," + = (""+num).substring(pos);
                }
                else
                    label =3D ""+num;
            }
            chartG.setColor(yAxisColor);
            if (!hideLeft)
    chartG.drawString(label, chartLeft - fm.stringWidth(label) - 2, z + =
```

```
fm.getDescent());
        chartG.drawString(label, chartRight + fm.stringWidth("9"), z + =
fm.getDescent());
                        if (lines)
                        {
                                chartG.setColor(scaleColor);
                                chartG.drawLine(chartLeft + 1, z, chartRight - 1, z);
                        }
                }
        }
        public void drawXAxis(boolean lines)
        {
                //DRAW X AXIS LABELS
                int steps =3D (int)(chartWidth / dateWidth);
                int xStep =3D (int)Math.max(0.9999 + (dateSpan / steps), 1);
                int xAxis =3D minDate;
                int xLabel =3D -1000;
                while ((xAxis < maxDate) && (xLabel < chartRight))
                {
                        if (dateToX(xAxis) > xLabel + dateWidth)
                        {
                                int x =3D dateToX(xAxis);
                                Date theDate =3D StockDetailAt(xAxis).getDate();
                String label =3D ""+(theDate.getMonth()+1) + "/" + theDate.getDate() =
+ "/" + theDate.getYear();
                chartG.setColor(xAxisColor);
                chartG.drawString(label, x + Math.max(((barWidth - =
fm.stringWidth(label)) >> 1), 0) , size().height - 3);  // x - =
(fm.stringWidth(label) >> 1)
```

```
                    if (lines)
                    {
                        chartG.setColor(scaleColor);
                        chartG.drawLine(x, chartTop+1, x, chartBottom-1);
                        if (showVolume)
                    chartG.drawLine(x, volumeTop+1, x, volumeBottom-1);
                    }
                    xLabel =3D x;
                }
                xAxis++;
            }
        }
        public int dateToX(Date d)
        {
            try
            {
                StockDetail temp =3D StockDetailAt(maxDate);
                if (temp.date.before(d))
                    return dateToX(maxDate+1);
                if (temp.date.equals(d))
                    return dateToX(maxDate);
                temp =3D StockDetailAt(minDate);
                if (temp.date.after(d))
                    return -1;//dateToX(minDate);
                for (int i =3D 0; i <=3D 99999; i++)
                {
                    StockDetail sd =3D StockDetailAt(i);
                    StockDetail sd2 =3D StockDetailAt(i+1);
                    if (sd.date.equals(d))
```

```
                        return dateToX(i);
                    if (sd2.date.equals(d))
                        return dateToX(i+1);
                    if (sd.date.before(d) && sd2.date.after(d))
                    {
                        return dateToX(i+1);
                    }
                }=09
            } catch (Exception e)=20
            {
                System.out.println(""+e);
            }
            return -1;
        }
        public Date indexToDate(int index)
        {
            try
            {
                StockDetail sd =3D StockDetailAt(index);
                return sd.date;
            }
            catch (Exception e) {}
            return null;
        }
        public int dateToX(int date)
        {
return chartLeft + (chartWidth * (date - minDate) / Math.max(dateSpan, = 1));
        }
        public int xToDate(int x)
```

```
        {
                return xToIndex(x);
        }

        public int valueToY(double value)
        {
                if (logChart)
                        return chartHeight - (int)((chartHeight * (Math.log(value) - =
Math.log(minValue)) / (Math.log(maxValue) - Math.log(minValue))));
                else
                        return chartHeight - (int)(chartHeight * (value - minValue) / =
Math.max(valueSpan, 0.0001));
        }
        public int openIntToY(double value)
        {
                return volumeHeight - (int)(volumeHeight * (value - =
currentHeader().minimumOpenint) / =
Math.max((currentHeader().maximumOpenint - =
currentHeader().minimumOpenint), 1)) + volumeTop;
        }
        public int percentToY(double per)
        {
                return chartHeight - (int)(chartHeight * (per - minPercent) / =
Math.max(percentSpan, 0.000001));
        }
        public int volumeToY(double value)
        {
        int retVal =3D volumeHeight - (int)(volumeHeight * (value - minVolume) =
/ Math.max(volumeSpan, 1)) + volumeTop;
```

```
            return retVal;
        }
        public double yToValue(int y)
        {
            if (logChart)
        return Math.pow(Math.E, ((Math.log(maxValue) - Math.log(minValue)) * =
(1 - ((double)y / (double)chartHeight)) + Math.log(minValue)));
            else
return ((y - chartHeight) * Math.max(valueSpan, 1)) / (-chartHeight) = + minValue;
        }
        public void paint(Graphics g)
        {
            g.drawImage(offImage, 0, 0, this);
        }
        public void update(Graphics g)
        {
            if (chartImage !=3D null)
                offG.drawImage(chartImage, 0, 0, this);
            if (haveData)
            {
                if (dragging)
                {
                                offG.setColor(Color.blue);
                                offG.setXORMode(Color.green);
                                offG.drawLine(mouseX, mouseY, dragX, dragY);
                                offG.dispose();
                                offG =3D offImage.getGraphics();
        =09
                } else {
```

```
if ((rightSelectedIndex > chartLeft) && (rightSelectedIndex < = chartRight) &&
(leftSelectedIndex > chartLeft) && (leftSelectedIndex < = chartRight))
                        {
                                offG.setColor(Color.blue);
                                offG.setXORMode(Color.green);
                                if (leftSelectedIndex < rightSelectedIndex)
                                {
        offG.fillRect(leftSelectedIndex, chartTop+1, rightSelectedIndex - =
leftSelectedIndex, chartHeight-1);
                offG.fillRect(leftSelectedIndex, volumeTop+1, rightSelectedIndex - =
leftSelectedIndex, volumeHeight-1);
                                }
                                else
                                {
                offG.fillRect(rightSelectedIndex, chartTop+1, leftSelectedIndex - =
rightSelectedIndex, chartHeight-1);
                offG.fillRect(rightSelectedIndex, volumeTop+1, leftSelectedIndex - =
rightSelectedIndex, volumeHeight-1);
                                }
                                offG.dispose();
                                offG =3D offImage.getGraphics();
                        } else {
                                if (cursor > 0)
                                {
                                        offG.setColor(Color.red);
        if ((leftSelectedIndex > chartLeft) && (leftSelectedIndex < = chartRight))
                                        {
offG.drawLine(leftSelectedIndex, chartTop, leftSelectedIndex, = chartBottom);
offG.drawLine(leftSelectedIndex, volumeTop, leftSelectedIndex, = volumeBottom);
```

```
if ((cursor =3D=3D 2) && (mouseY >=3D chartTop) && (mouseY <=3D =
chartBottom))
                        offG.drawLine(chartLeft, mouseY, chartRight, mouseY);
                                }
                        }
                        if ((currentHeader().splits !=3D null) && !usePercent)
                                {
        StringTokenizer st =3D new = StringTokenizer(currentHeader().splits, ",");
                                        while (st.hasMoreTokens())
                                        {
                                Date d =3D new Date(st.nextToken());
                                                String f =3D st.nextToken();
                                int x =3D dateToX(d) + (barWidth >> 1);
                                                int w =3D 5;
                                if (x =3D=3D leftSelectedIndex)
                                                {
                                        offG.setFont(labelFont);
                                                int x1 =3D x - 50;
                                                int y1 =3D mouseY + 24;

offG.setColor(Color.yellow);
offG.fillRect(x1, y1, 90, chartG.getFontMetrics().getHeight() + = 4);

offG.setColor(Color.black);
offG.drawRect(x1, y1, 90, chartG.getFontMetrics().getHeight() + = 4);
                                        int factor =3D Integer.parseInt(f, 10);
                                        String s =3D "";
                                                switch (factor)
                                                {
```

```
                                                    case 50:
            s =3D "2 for 1"; break;          =09
                                                    case 33:
            s =3D "3 for 1"; break;=09
                                                    case 66:
            s =3D "3 for 2"; break;
                                                    case 25:
            s =3D "4 for 1"; break;
                                                    case 75:
            s =3D "4 for 3"; break;
                                                    case 20:
            s =3D "5 for 1"; break;
                                                    case 60:
            s =3D "5 for 3"; break;
                                                    case 80:
            s =3D "5 for 4"; break;
                                                    case 83:
            s =3D "6 for 5"; break;
                                                    case 10:
            s =3D "10 for 1"; break;
                                                    case 1:
                  s =3D "100 for 1"; break;
                                                    case 600:
            s =3D "1 for 6"; break;
                                                    }
      offG.drawString("Split: " + s, x1 + 4, y1 + = chartG.getFontMetrics().getHeight());
                                                }
                                            }
                                        }
```

```
                    }
                }
            =09
                offG.setFont(titleFont);
                offG.setColor(Color.gray);
                if (currentHeader() !=3D null)
                {
                    legend =3D currentHeader().symbol;
                    if (!showSymbol)
                        legend =3D "";
                if ((currentHeader().companyName !=3D null) && (showSymbol))
                        legend +=3D " - ";
                    if (currentHeader().companyName !=3D null)
                    legend =3D legend + currentHeader().companyName;
                    if (currentHeader().duration >=3D 4)
                        legend +=3D " [monthly]";
                    else if (currentHeader().duration >=3D 2)
                        legend +=3D " [weekly]";
offG.drawString(legend, chartLeft+10, chartTop + = chartG.getFontMetrics().getHeight()
+ 6);
                }
            } else { //no data yet
                offG.setFont(titleFont);
                offG.setColor(Color.black);
                offG.drawString("Ready to Chart", chartLeft+10, chartTop + =
chartG.getFontMetrics().getHeight() + 6);
                offG.setFont(labelFont);
            =09
                offG.drawString("This site is free. If you accept the terms of =
```

```
Prophet's User Agreement, ", chartLeft+10, chartTop + =
chartG.getFontMetrics().getHeight() + 30);
                offG.drawString("enter a symbol and press <Enter>", chartLeft+10, =
chartTop + (chartG.getFontMetrics().getHeight() * 2) + 30);
        }
        paint(g);
    }
    public void moveLine(int x, int y)
    {
        int index =3D xToIndex(x);
        if (index !=3D currentIndex)
        {
            leftSelectedIndex =3D dateToX(index) + (barWidth >> 1);
            currentIndex =3D index;
            repaint();
        }
        if (index >=3D0)
        {
            StockDetail sd =3D StockDetailAt(index);
            setDetail(sd, yToValue(y));
            repaint();
        }
    }
    public void moveLineIndex(int x)
    {
        int index =3D x;
        if (index !=3D currentIndex)
        {
            leftSelectedIndex =3D dateToX(index) + (barWidth >> 1);
```

```
                currentIndex =3D index;
                repaint();
            }
            if (index >=3D0)
            {
                StockDetail sd =3D StockDetailAt(index);
                setDetail(sd, -1);
            }
        }
        public int xToIndex(int x)
        {
            try
            {
                for (int i =3D minDate; i <=3D maxDate; i++)
                {
                    StockDetail sd =3D StockDetailAt(i);
                    if ((sd.x1 <=3D x) && (sd.x2 >=3D x))=20
                        return i;
                }=09
            } catch (Exception e) {}
            return -1;
        }
        public StockDetail StockDetailAt(String s, int i)
        {
            StockHeader sh =3D getHeader(s);
            if (sh =3D=3D null)
                return null;
            return sh.dataAt(i);
        }
```

```
public StockDetail StockDetailAt(int i)
{
        if (currentHeader() !=3D null)
                return currentHeader().dataAt(i);
        return null;
}
public boolean mouseMove(Event evt, int x, int y)
{
        moveLine(x, y);
        mouseY =3D y;
        return true;
}
=09
public boolean mouseDrag(Event evt, int x, int y)
{
        /*if (x > size().width)
        {
                dragX =3D -1;
                dragY =3D -1;
                mouseX =3D -1;
                mouseY =3D -1;
                dragging =3D false;
                repaint();
                return true;
        }*/
 if (((evt.modifiers & Event.SHIFT_MASK) !=3D 0) && (mouseX >=3D 0))
        {
                dragging =3D true;
                dragX =3D x;
```

```
                dragY =3D y;
                repaint();
            } else {
                rightSelectedIndex =3D x;
                if (rightSelectedIndex < chartLeft)
                    rightSelectedIndex =3D chartLeft + 1;
                if (rightSelectedIndex > chartRight)
                    rightSelectedIndex =3D chartRight - 1;
        =09
                int index =3D xToIndex(rightSelectedIndex);
                if (index >=3D0)
                {
                    StockDetail sd =3D StockDetailAt(index);
                    setRightDetail(sd);
                }
                repaint();
            }
            return true;
        }
        public boolean mouseEnter(Event evt, int x, int y)
        {
            mouseInView =3D true;
            leftSelectedIndex =3D x;
            moveLine(x, y);
            return true;
        }
        public boolean keyDown(Event e, int key)=20
        {
            if (key =3D=3D 1006)  //left
```

```
{
        if (currentIndex <=3D minDate + 1)
        {
                int move =3D Math.max(minDate - (dateSpan >> 1), 0);
                move =3D minDate - move;
                minDate -=3D move;
                maxDate -=3D move;
                updateChart();
        }
        if (currentIndex > 0)
                moveLineIndex(currentIndex - 1);
        return true;
}
if (key =3D=3D 1007)  //right
{
        if (currentIndex >=3D maxDate - 1)
        {
                int move =3D maxDate + (dateSpan >> 1);
                if (move > currentHeader().count())
                        move =3D currentHeader().count() - 1;
                move =3D move - maxDate;
                if (move =3D=3D 0) move =3D 1;
                if (move =3D=3D -1) move =3D 0;
                minDate +=3D move;
                maxDate +=3D move;
                //dateSpan =3D maxDate - minDate;
                updateChart();
        }
        if (currentIndex < currentHeader().count() - 1)
```

```
                moveLineIndex(currentIndex + 1);
            return true;
        }
        postEvent(new Event(this, SYMBOL_CHANGED, "" + (char)key));
        return false;
    }
    public boolean mouseDown(Event evt, int x, int y)
    {
        requestFocus();
        if ((evt.modifiers & Event.META_MASK) !=3D 0)
        {
            if ((evt.modifiers & Event.SHIFT_MASK) !=3D 0)
            {
                drags.removeElementAt(drags.size() - 1);
                updateCookie();
                updateChart();
            } else if ((evt.modifiers & Event.CTRL_MASK) !=3D 0)
            {
                drags.removeAllElements();
                updateCookie();
                updateChart();
            }
            else
                unZoom();
        }
        mouseX =3D x;
        mouseY =3D y;
        return true;
    }
```

```
=09
        public boolean mouseUp(Event evt, int x, int y)
        {
                if (dragging)
                {
                        int rightDate =3D xToDate(dragX);
                        if (rightDate =3D=3D -1)
                        {
                                if (dragX > chartRight)
                                {
                                        rightDate =3D maxDate;
        int sh =3D (mouseY - dragY) * (chartRight - mouseX) / (dragX - = mouseX);
                                        dragY =3D mouseY - sh;
                                }
                                if (dragX < chartLeft)
                                {
                                        rightDate =3D minDate;
int sh =3D (mouseY - dragY) * (mouseX - chartLeft) / (mouseX - = dragX);
                                        dragY =3D mouseY - sh;
                                }
                        =09
                        }
ADrag ad =3D new ADrag(xToDate(mouseX), yToValue(mouseY), rightDate, =
yToValue(dragY));
                        drags.addElement(ad);
                        updateCookie();
                        updateChart();
                } else {
                        int leftIndex =3D xToIndex(leftSelectedIndex);
```

```
                int rightIndex =3D xToIndex(rightSelectedIndex);
        if ((leftIndex >=3D0) && (rightIndex >=3D 0) && (Math.abs(leftIndex - =
rightIndex) > 4))
                {
                        if =
(StockDetailAt(leftIndex).getDate().after(StockDetailAt(rightIndex).getDa=
te()))
                                setDates(StockDetailAt(rightIndex).getDate(), =
StockDetailAt(leftIndex).getDate());
                        else
                                setDates(StockDetailAt(leftIndex).getDate(), =
StockDetailAt(rightIndex).getDate());
                }
                rightSelectedIndex =3D -1;
                removeRightDetail();
                moveLine(x, y);
        }
        dragging =3D false;
        rightSelectedIndex =3D -1;
        return true;
    }
    public boolean mouseExit(Event evt, int x, int y)
    {
        mouseInView =3D false;
        dragging =3D false;
        mouseX =3D -1;
        removeRightDetail();
        moveLine(x, y);
        return true;
```

```
}
public void removeDrags()
{
        drags.removeAllElements();
        updateCookie();
}
public StockHeader deleteHeader(String s)
{
        int i =3D 0;
        while (i < stockHeaders.size())
        {
                if
(((StockHeader)stockHeaders.elementAt(i)).symbol.equals(s))=20
                {
                        stockHeaders.removeElementAt(i);
                }
                i++;
        }
        return null;
}
public StockHeader getHeader(String s)
{
        int i =3D 0;
        while (i < stockHeaders.size())
        {
                if
(((StockHeader)stockHeaders.elementAt(i)).symbol.equals(s))=20

                {
```

```
                    return (StockHeader)stockHeaders.elementAt(i);
                }
                i++;
            }
            return null;
        }
        public StockHeader getHeader(String s, int d)
        {
            int i =3D 0;
            while (i < stockHeaders.size())
            {
            if ((((StockHeader)stockHeaders.elementAt(i)).symbol.equals(s)) &&
            (((StockHeader)stockHeaders.elementAt(i)).duration =3D=3D d))
                {
                    return (StockHeader)stockHeaders.elementAt(i);
                }
                i++;
            }
            return null;
        }
        public StockHeader currentHeader()
        {
            return getHeader(symbol);
        }
        public void loadStock(String sym, String name, boolean setSymbol, =
    boolean ss, int duration)
        {
            showSymbol =3D ss;
            loadStock(sym, name, setSymbol, duration);
```

```
}
public void loadStock(String sym, String name, boolean setSymbol, int = duration)
{
    forceScale =3D false;
    sym =3D sym.toUpperCase();
    System.out.println("Loading " + sym);
    if (getHeader(sym, duration) !=3D null)
    {
        if (setSymbol)
        {
            symbol =3D sym;
        postEvent(new Event(this, SYMBOL_CHANGED, symbol));
        }=09
        unZoom();
        drags.removeAllElements();
        cookieToGet =3D sym;
        updateChart();
        unZoom();
        requestFocus();
        return;
    }
    deleteHeader(sym);
    if (setSymbol)
    {
        postEvent(new Event(this, SYMBOL_CHANGED, "Loading..."));
        haveData =3D false;
    }
    setMouseCursor(Frame.WAIT_CURSOR);
    String inline;
```

```
String input =3D "";
String companyName =3D null;
Graphics g =3D this.getGraphics();
String splitInput =3D null;
try=20
{
        g.setColor(Color.blue);
g.drawString("Connection to server", chartLeft + 5, chartBottom - 6);
        g.setColor(Color.white);
g.fillRect(chartLeft+1, chartBottom-20, (int)(chartWidth * 1-2), 20);

        URL   inputURL =3D new URL(parent.getCodeBase(), script +
sym + = "&d=3D" + duration);
        URLConnection       inputConnection =3D
inputURL.openConnection();
        inputConnection.setDefaultRequestProperty("CONTENT_TYPE",
=
"application/x-www-form-urlencoded");
        DataInputStream dis =3D new =
DataInputStream(inputConnection.getInputStream());
        int count =3D 0;
        chartG.setFont(new Font("Dialog", Font.PLAIN, 8));
        double per =3D 0;
        boolean splits =3D false;
        while ((inline =3D dis.readLine()) !=3D null)=20
        {
                count++;
                per =3D Math.min(1, count / 180.0);
                g.setColor(Color.blue);
```

```
g.fillRect(chartLeft+1, chartBottom-20, (int)(chartWidth * per) - 2, = 20);
                        g.setColor(Color.white);
        g.drawString("Loading " + ((name !=3D null) ? name : sym) + ": " + =
(int)(100*per) + "%", chartLeft + 5, chartBottom - 6);
                        if (inline.startsWith("SPLITS"))
                        {
                                splits =3D true;
                                splitInput =3D "";
                        } else {
                                if (splits)
                                        splitInput +=3D inline + ",";
                                else
                                        input +=3D inline + "\n";
                        }
                }
                dis.close();

                g.setColor(Color.blue);
        g.fillRect(chartLeft+1, chartBottom-20, (int)(chartWidth * per) - 2, =
20);
                g.setColor(Color.white);
                if (name =3D=3D null)
                {
        g.drawString("Loading company name", chartLeft + 5, chartBottom - =
6);
                inputURL =3D new URL(parent.getCodeBase(), nameScript + sym);
                        inputConnection =3D inputURL.openConnection();
                        dis =3D new
DataInputStream(inputConnection.getInputStream());
```

```
                companyName =3D dis.readLine();
                g.setColor(Color.blue);
        g.fillRect(chartLeft+1, chartBottom-20, (int)(chartWidth * 1), 20);
        } else
                companyName =3D name;
        postEvent(new Event(this, ADD_RECENT, sym));
}
catch (Exception e) {=20
        postEvent(new Event(this, SYMBOL_CHANGED, "no data"));
        haveData =3D false;
        System.out.println(""+e.toString());=20
}
g.setColor(Color.blue);
g.fillRect(chartLeft+1, chartBottom-20, (int)(chartWidth * 1), 20);
g.setColor(Color.white);
g.drawString("Done", chartLeft + 5, chartBottom - 6);
if (input.startsWith("ERROR"))
{
        if (setSymbol)
        {
        postEvent(new Event(this, SYMBOL_CHANGED, "no data"));
        }
        haveData =3D true;
} else {
        if (setSymbol)
        {
                symbol =3D sym;
        postEvent(new Event(this, SYMBOL_CHANGED, symbol));
                haveData =3D true;
```

```
                    }
                    StockHeader sh =3D new StockHeader(sym, companyName, input,
=
splitInput, duration);
                    stockHeaders.addElement(sh);
                    unZoom();
                    drags.removeAllElements();
                    cookieToGet =3D sym;
                    updateChart();
                    unZoom();
                    requestFocus();
                }
                setMouseCursor(Frame.DEFAULT_CURSOR);
            }
    public void setMouseCursor(int type)
    {
        Object frame =3D new Object();
        for( frame =3D ((Component) this).getParent(); !( frame =
instanceof Frame ); frame =3D ((Component) frame).getParent());
        ((Frame) frame).setCursor(type);
    }

        public void setDetail(StockDetail sd1, double d)
        {
                currentDetail =3D sd1;
                currentValue =3D d;
                postEvent(new Event(this, DETAIL_CHANGED, null));
        }
        public void removeRightDetail()
```

```
{
        postEvent(new Event(this, REMOVE_DETAIL, null));
}
public void setRightDetail(StockDetail sd1)
{
        postEvent(new Event(this, RIGHTDETAIL_CHANGED, sd1));
}
public String getSymbol()
{
        return symbol;
}
public void setTrendCookie(String val)
{
        cookieToGet =3D null;
        try
        {
                if ((val =3D=3D null) || (val.equals("")))
                        return;
        =09
                StringTokenizer st =3D new StringTokenizer(val, ";");
                while (st.hasMoreTokens())
                {
        StringTokenizer st2 =3D new StringTokenizer(st.nextToken(), ",");
        int d1 =3D xToIndex(dateToX(new Date(st2.nextToken())));
        double v1 =3D new Double(st2.nextToken()).doubleValue();
        int d2 =3D xToIndex(dateToX(new Date(st2.nextToken())));
        double v2 =3D new Double(st2.nextToken()).doubleValue();
                        ADrag ad =3D new ADrag(d1, v1, d2, v2);
                        drags.addElement(ad);
```

```
                    }
                    updateChart();
                }
                catch (Exception e)
                {}
        }
=09
        public void updateCookie()
        {
                cookieValue =3D null;
                String newCookie =3D "";
                for (int i =3D 0; i < drags.size(); i++)
                {
                        ADrag ad =3D (ADrag)drags.elementAt(i);
                        if ((indexToDate(ad.getDate1()) !=3D null) &&
                                (indexToDate(ad.getDate2()) !=3D null))
                        {
                        newCookie +=3D shortDate(indexToDate(ad.getDate1())) + ",";
                                newCookie +=3D
("+ad.getValue1()+"000000").substring(0, 5) + ",";
                        newCookie +=3D shortDate(indexToDate(ad.getDate2())) + ",";
                                newCookie +=3D
("+ad.getValue2()+"000000").substring(0, 5) + ";";
                        }
                }
                cookieValue =3D newCookie;
        }

        public static String shortDate(Date d)
```

```
{
        return "" + (d.getMonth() + 1) + "/" +
                d.getDate() + "/" +
                (d.getYear() + 1900);
}
public void setLog(boolean b)
{
        logChart =3D b;
        forceScale =3D true;
        updateChart();
}
public void setCursor(int i)
{
        cursor =3D i;
        repaint();
}
public void checkRepaint()
{
        if (currentSize !=3D size().width * 1000 + size().height)
        {
                unZoom();
                currentSize =3D size().width * 1000 + size().height;
                System.out.println("repainting");
        }
}
public void print()
{
        try
        {
```

```
                Graphics g =3D this.getGraphics();
                g.setColor(Color.black);
        g.drawString("Creating Printable Image", chartLeft + 5, chartBottom - = 6);
                Image testImage =3D createImage(size().width, size().height);
                Graphics testG =3D testImage.getGraphics();
                testG.drawImage(offImage, 0, 0, this);
                testG.setFont(titleFont);
                testG.setColor(Color.gray);
    testG.drawString(legend, chartLeft+10, chartTop + = chartG.getFontMetrics().getHeight()
    + 6);
                MediaTracker mt =3D new MediaTracker(this);
                mt.addImage(testImage, 0);
                mt.waitForAll();
                URL printURL =3D new URL(parent.getCodeBase(), saveScript);
                URLConnection connect =3D printURL.openConnection();
                connect.setDoOutput(true);  =20
                PrintStream ps =3D new PrintStream(connect.getOutputStream());
                ByteArrayOutputStream ba =3D new ByteArrayOutputStream();
                new GifEncoder(testImage, ba).encode();
                byte buf[] =3D ba.toByteArray();
                ps.print("image=3D");=20
                String s;
                for (int i =3D 0; i < buf.length; i++)
                {
                        s =3D Integer.toHexString(buf[i]);
                        if (s.length() =3D=3D 1)
                                s =3D "0" + s;
                        else if (s.length() > 2)
                                s =3D s.substring(s.length() - 2);
```

```
                    ps.print("%" + s);
                    if (i % 10 =3D=3D 0)
                    {
                double per =3D Math.min(1, (double)i / (double)buf.length);
                        g.setColor(Color.blue);
g.fillRect(chartLeft+1, chartBottom-20, (int)(chartWidth * per) - = 2, 20);
                        g.setColor(Color.white);
g.drawString("Creating Printable Image", chartLeft + 5, chartBottom = - 6);
                    }
                }
                ps.print("&jody=3Dpowlette");
                System.out.println("done");
                ps.close();          =20
DataInputStream dis =3D new = DataInputStream(connect.getInputStream());
                String inline;
                inline =3D dis.readLine();
                if (inline.startsWith("SUCCESS"))
                {
                    inline =3D dis.readLine(); //get url
                    System.out.println(inline);
                    URL url =3D new URL(inline);
                    parent.getAppletContext().showDocument(url, "_blank");
                } else {
                    System.out.println("error: " + inline);
                    while ((inline =3D dis.readLine()) !=3D null)
                        System.out.println(inline);
                }
```

```
                }
                catch (Exception e)
                {
                        System.out.println("Printing Error: " + e);
                }
                repaint();
        }
}

Content-Type: application/octet-stream;
        name="ImageSave.pl"
Content-Transfer-Encoding: quoted-printable
Content-Disposition: attachment;
        filename="ImageSave.pl"
require "cgi-lib.pl";
&ReadParse(*input);
$imgdir =3D "d:/inetpub/prophetcharts/graphs";
$urldir =3D "http://www.prophetcharts.com/graphs/";
print "Content-type: text/html\n\n";
$i =3D 0;
while (-e "$imgdir\\chart$i.gif") {
        $i++;
}
open (OUTFILE, ">$imgdir\\chart$i.gif");
binmode(OUTFILE);
print OUTFILE $input{'image'};
close (OUTFILE);
print "SUCCESS\n";
print "http://www.prophetcharts.com/printDoc.asp?id=3D$i";
```

```
opendir(DIR, $imgdir);
@files =3D readdir(DIR);   =20
closedir DIR;
foreach $file (@files)
{
        if ($file =3D~ /\.gif$/)=20
        {
                =
($dev,$ino,$mode,$nlink,$uid,$gid,$rdev,$size,$atime,$mtime,$ctime,$blksi=
ze,$blocks)  =3D stat("$imgdir/$file");        =09
                if (time - $mtime > 10 * 60)=20
                {
                        unlink "$imgdir/$file";
                }
        }
}
```